(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
13 December 2001 (13.12.2001)

PCT

(10) International Publication Number
WO 01/95116 A2

(51) International Patent Classification[7]: G06F 13/00, H04L 12/50

(21) International Application Number: PCT/DK01/00389

(22) International Filing Date: 7 June 2001 (07.06.2001)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
60/209,801 7 June 2000 (07.06.2000) US

(71) Applicant *(for all designated States except US)*: **NETBLOOM A/S** [DK/DK]; Fuglevangsvej 9, DK-1962 Frederiksberg C (DK).

(72) Inventor; and
(75) Inventor/Applicant *(for US only)*: **JOHANSEN, Ulf** [DK/DK]; Nitivej 18, 1.tv, DK-2000 Frederiksberg (DK).

(74) Agent: **CHAS. HUDE A/S**; H.C. Andersens Boulevard 33, DK-1780 Copenhagen V (DK).

(81) Designated States *(national)*: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DZ, EC, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, TZ, UA, UG, US, UZ, VN, YU, ZA, ZW.

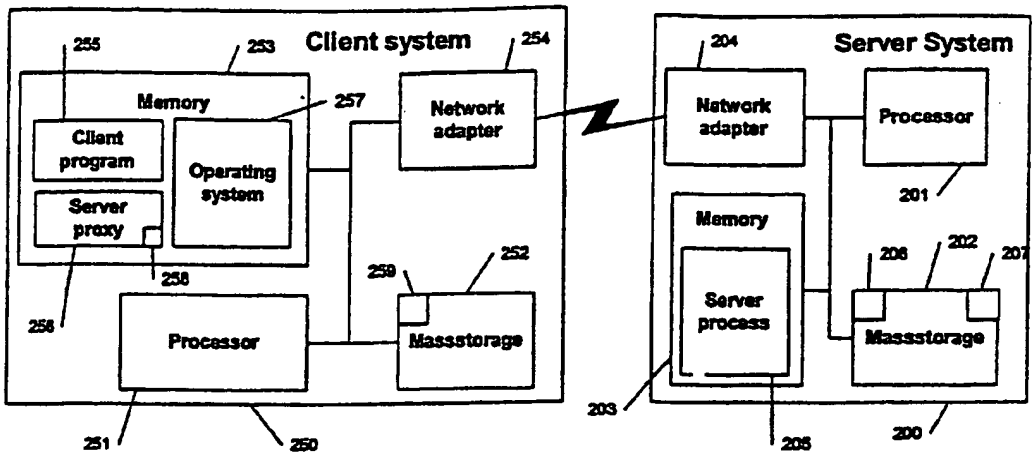
(84) Designated States *(regional)*: ARIPO patent (GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE, TR), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GW, ML, MR, NE, SN, TD, TG).

Declaration under Rule 4.17:
— *of inventorship (Rule 4.17(iv)) for US only*

Published:
— *without international search report and to be republished upon receipt of that report*

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

(54) Title: A METHOD AND A SYSTEM FOR PROVIDING INFORMATION FROM A SERVER TO A CLIENT



(57) Abstract: A method of only transmitting to a first computer the full information required from a second computer the first time is described, where, when the same information is required again, processor controlling code having been transmitted from the second computer to the first computer is executed. This code may then have the first computer perform specific tasks such as storing or retrieving information at the first computer thus reducing the amount of information transmitted during subsequent the request for information.

WO 01/95116 A2

Title: A method and a system for providing information from a server to a client

Technical field

The present invention relates to a method and a system for providing information from a server to a client. More particularly, this invention provides an improved system for caching resources on a client using resource templates containing executable instructions in a client-server framework.

The invention aims at reducing the amount of data being transmitted between a server process and a client process in a client-server framework, more particularly providing a improved system for caching resources on the client using resource templates containing executable instructions in the client-server framework.

Background Art

Many computer systems, including systems using the Internet such as the World Wide Web, are based on the conventional client-server system architecture model. The term "client" and "server" are used to refer to a computer's general role as requester of data (the client) and the provider of data (the server).

A client-server application comprises a "client process" located on the client and a "server process" located on the server. In the client-server model the client and the server share a common agreement (often called a communication protocol or simply "protocol") of how communication between them is conducted.

The nature of these protocols varies, but typically the client issues a "request" that contains information identifying a piece of data located on the server or a specific operation to be performed on or by the server. The server receives the request, performs whatever processing is needed to produce the data or perform the operation and

returns a "response" to the client containing the requested data or the result of the performed operation.

Network bandwidth and server processing resources are often limited, which makes it important to optimize the usage of these resources in client-server frameworks.

A common optimization is to cache the responses from the server. When requesting a piece of data from a server, the client may store (cache) a temporary copy of the response. This copy is then used the next time the same data is requested rather than re-transmitting the request to the server. Using timestamps and a specific set of rules, often defined in the protocol, the client can avoid using a cached resource, if an updated version of the resource is present on the server.

In addition, the caching may be performed by an intermediate computer on a network path from the client to the server, using the same principle but sharing cached resources between multiple users accessing the server.

When resources on the server change frequently, the optimization impact of a simple caching scheme is reduced, in worst case vanishing.

This is especially problematic if only a small fraction of a particular resources changes often, while a greater part of the resource change infrequently and the client, for some reason, is incapable of requesting the data that change often and the data that change infrequently separately, as the overall result will be that all the data will fail to cache properly.

Under the Web environment, Web browsers (the client processes) reside in the client and specially formatted HTML documents, images and other types or media reside on Internet Web servers (the server processes). Web clients and Web servers typically communicate using the "HyperText Transfer Protocol" (HTTP).

The browser retrieves resources from the server by sending a request to the Web server identifying the specific resource using a "Uniform Resource Locator" (URL).

HTTP and web browsers support a caching scheme for caching the resources retrieved from the web server, including a set of rules which ensure that the web server can control how resources are cached in order to avoid reusing stale resources. Likewise, proxy servers functioning as multi-user caches are widely used to improve the performance of browser web sites.

However, modern web sites are becoming increasingly dynamic, and rather than being simple document servers, the role of the web servers are more often than not that of a traditional application server.

HTML documents serve as the user interface for the client-server application and are dynamically generated specifically for the particular user and context of operation. These documents often contain large amount of data that change infrequently while small data specific parts of the documents change frequently. Such a page cannot be cached by the client or by proxy servers because the page as a whole is changed.

It is desirable to improve the caching capabilities in this particular client-server framework.

Also, it is desired to overcome the shortcomings in client-server frameworks where traditional caching is insufficient to obtain optimal usage of bandwidth and server resources by making is difficult or impossible to avoid data or parts of data to be repeatedly retransmitted from the server to the client. The content assembly server proxy 256 according to the invention allows e.g., installing or caching, on the client, resource templates that may include scripts updating only the parts of the resource that are changed when accessed by the client.

Summary of the invention

In one aspect, the invention relates to a method for providing information at a first computer. The method includes:

- at a first point in time:
  - providing, at the first computer, information identifying the information desired,
  - transferring the identifying information from the first computer to a second computer,
  - transferring processor executable computer code from the second computer to the first computer,
- at a point in time subsequent to the first point in time:
  - providing, at the first computer, the identifying information,
  - executing in or at the first computer the computer code.

Thus, instead of simply re-requesting the information from the second computer the second or subsequent times, and instead of simply retrieving cached information from a local cache or proxy server, the computer code is executed which provides a new range of possibilities in the retrieving of the information. In the present context, processor executable computer code will mean code or instructions which are adapted to control a controllable processor. Normally, the processor will be a general purpose processor and the code assembler code therefor or code in any high level language.

Information identifying the information may be any type of information, such as an address or location thereof. An address or location may be any address or location on any computer, normally the second computer. Today, a large amount of information is available on the Internet where home pages or Internet pages have their own addresses - and all resources have their own Universal Resource Locators, URL's that point directly to the individual resources. It is not important how or where these pages or resources are positioned in relation to the first computer, due to the fact that the

Internet search machines are fully adapted to locate the information required. Thus, the location or address may be on a single computer and be specified as a drive/disc thereof and an identifier of a file thereon - or it may in the other extreme simply be a unique address which is sought for on the Internet.

Transferring information or code may be done in any manner suitable, such as wireless, by wire, using modems, ISDN, or any other manner of data transport. In one embodiment, at the first point in time, information is transferred from the second computer to the first computer, part of the transferred information being stored locally at or in the first computer.

Thus, the execution of the computer code may comprise retrieving the part of the information from the local store and requesting, from the address or location, one or more other parts cf the information. Especially in the situation where part of the information required is often updated and other parts thereof quite rarely, the rarely updated information may be stored locally and the often-updated parts retrieved "on line". In this situation, the execution of the computer code might comprise determining whether information required is stored in the local store. In this embodiment, at the first point in time, information identifying the stored information may be stored at or in the first computer - optionally together with information relating to the address or location - in order for the first computer to determine, at the second point in time, that stored information relates to the information required.

One manner of realizing this operation is to provide at or in the first computer a list of addresses or locations the corresponding computer code has been received and is stored in or at the first computer.

Another functionality is one wherein the execution of the computer code comprises:

- providing, at the first computer, information relating to a second address or location, and

- transferring information from the second address or location to the first computer.

Thus, additional information may be provided from an address or location different from that which initially was used for requesting the information required. This may be useful when the information relates, e.g., to home pages where the initial address or location would be the Internet address of the page. However, in order to re-route users of this specific bandwidth saving method e.g., to another server, the executing computer code may re-direct the subsequent request for the home page to another address or server having another functionality - such as not transmitting the full contents of the home page if it knows that part thereof will already be stored on the first computer.

In this situation, at the first point in time, part of the information may be stored locally at or in the first computer and, at the subsequent point in time, the part of the information may be retrieved from the local store and one or more other parts of the information is transferred from the second address or location to the first computer.

In general, when part of the received information is stored locally, the execution of the computer code may comprise determining whether information stored in the local store is outdated and, in that situation, requesting updated information from the second computer.

Also, in that situation, the execution of the computer code may comprise storing at least part of the transferred information locally at or in the first computer or retrieving information therefrom, the storing or retrieval being performed in a local database defined and/or maintained by execution of processor executable computer code transferred from the second computer.

Thus, the execution of the computer code may actually form a "remote controlled" database in or at the first computer. This database may be fully defined and maintained by the execution of the computer code in order for the programmer thereof to have the full control thereof. In addition, the execution of the computer code may make it possible for that programmer to also cease control of other parts of the first computer, such as external units (CD readers/writers, printers, I/O-ports, etc.) and internal units, such as individual drives on hard discs, etc. The information stored locally may be resource templates, which normally rarely vary. More likely, individual resources to be "filled there into" will vary often.

The process of selecting the resource template may comprise the processes of defining a namespace by associating the identity of a resource on the server or a set of identities with a local resource template or group of templates, retrieving the identity of the requested information from the request and in case this identity is associated with a resource template, selecting the resource template.

In the present context, a name space is a denotation defined by the computer code and which relates to a storing location or another resource locally on the first computer - such as in the database or a disc drive thereof. Preferably, all first computers will have the same denotations for the same name spaces - even though the actual positions of the individual databases may vary (such as positioned under different directories). Most preferably, also the positions of the databases and other parts related to name spaces will be the same in the first computers.

For example, the process of selecting the resource template for a may comprise the steps of defining a namespace by associating a specific URL or a subset of URL's sharing a common base URL with a file or directory in the local file system, retrieving the URL from the request and in case the URL matches a namespace, selecting the resource template from the local file system.

Alternatively, the process of selecting the resource template may comprise the steps of downloading a temporary resource template from the server when the resource is initially requested and caching the resource template for subsequent requests, retrieving the resource identity from subsequent requests and in case the identity matches a cached resource template, selecting the resource template from the list of cached resource templates.

The present invention may be provided as an add-on to a standard browser, where the add-on, in accordance with one aspect of the present invention, a method of assembling content, comprises the processes of intercepting the request from the client, selecting a local resource template if one is available to the server proxy 256 object, process any scripts in the resource template and deliver the resource to the client. The client process may be a web browser and the server process may be a web server. The system consistent with the present invention may be implemented in a way which makes it transparent for the client process that the server proxy 256 handles the request and not the actual server to which the request was made. In addition, the server proxy 256 may be implemented coincident and cooperatively with existing caching systems, such as for example the cache functionality in a web browser.

In one situation, the information relating to the address or location is a URL. The information to be, provided may be a home page, a web site, or one or more parts thereof, where the information comprises a number of independent resources, and wherein the address or location thereof is on or at a second computer holding the independent resources.

In this situation, the method may comprise identifying, at the first computer, parts of the information which are present at the first computer,

- requesting, at the second computer, parts of the information not present at the first computer,
- retrieving the information required stored in or at the first computer,

- assembling/combining the information required and providing it to a user of the first computer such as on a monitor or a printer thereof.

Normally, the assembling of the information comprises of the steps of post processing requests as part of the network functionality of a client application, selecting a local resource template if one is available to the server proxy 256 object, process any scripts in the resource template and deliver the resource to the client. Executing scripts in the resource template may comprise the steps of scanning the resource template for special marks identifying scripts, executing the set of instructions within these marks, replacing the script blocks with the output resulting from executing the set of instructions in the script block.

Another functionality is one where the process of transferring the information relating to the address or location to the second computer comprises transmitting additional information to the second computer, and wherein the process of transmitting the processor executable computer code to the first computer is conditioned on the second computer receiving the additional information, the second computer otherwise transmitting the information required to the first computer. In this situation, the first computer acknowledges, by the presence of the additional information, that it is adapted to perform the present method, whereby the second,computer transmits the computer code. Computers requesting the same information without the additional information will simply receive the information requested.

An interesting embodiment is one in which:

in each of a plurality of first computers:

- at a respective first point in time:
  - providing, at the first computer, information relating to an address or location of the information desired,
  - transferring the information relating to the address or location from the first computer to a second computer,

- transferring processor executable computer code from the second computer to the first computer,
- at a respective point in time subsequent to the respective first point in time:
  - providing, at the first computer, the information relating to the address or location,
- executing in or at the first computer the computer code , where the information relating to the addresses or locations provided relates to the same information and wherein the processor executable computer code transferred is adapted to provide the same effect in the first computers when executed (the same).

Thus, a plurality of first computers firstly request the information, receive the computer code and executes the computer code the next time the information is requested. Naturally, the first point in time does not have to expire at the same time for all first computers.

In this manner, similar or the same computer code is transmitted to all first computers, whereby, upon or during execution, the same effect is obtained. It should be noted that the computer code transmitted may vary from first computer to first computer due to a number of varying factors, such as operating system, computer contents or performance. However, the overall result of the execution of the computer code should be the same (storage and retrieval of information, deletion of information, output of the computer, etc.).

Thus, in each first computer at the respective first point in time, information is preferably transferred from the second computer to the first computer, the same parts of the transferred information being stored locally at or in each first computer. Then, the execution of the computer code may comprise, in each first computer, retrieving the parts of the information from the local store and requesting, from the address or location, the same one or more other parts of the information.

A second aspect of the invention relates to a system for providing information from a first computer to a second computer, the system comprising a first and a second computer where: the first computer comprising:

- a mechanism constructed and adapted to provide information relating to an address or location of the information desired,
- a mechanism constructed and adapted to transfer the information from the address or location to the first computer to the second computer,
- a mechanism constructed and adapted to receive, from the second computer, processor executable computer code, and
- a mechanism constructed and adapted to execute the computer code,
- a mechanism constructed and adapted to determine whether the address or location is provided by the providing a mechanism constructed and adapted to the first or a subsequent time and for, if the address or location is provided a subsequent time, operating the executing means to execute the computer code,

the second computer comprising:

- a mechanism constructed and adapted to receive the address or location,
- a mechanism constructed and adapted to transmit the computer code to the first computer.

In one embodiment,

- the first computer comprises storage,
- the receiving means comprises a mechanism constructed and adapted to receive information and for storing at least part of the received information in the storage,

where the executing means being adapted to retrieve at least part of the stored information when executing the computer code.

Especially where the executing means are adapted to, while executing the computer code, define and/or maintain a database held by the storage means, and wherein the storing means comprise a mechanism constructed and adapted to provide information

relating the address or location with a location of the stored information, advantages are seen - also when a number of first computers are "controlled" by the same server in the same manner.

The executing means may be adapted to, while executing the computer code, determine whether all or part of the stored information to be retrieved is outdated and to, in that situation, request updated information from the second computer. Due to the fact that this determination is performed by the execution of the computer code, any desired timing or requirements for refreshing information may be used-and different requirements for each piece of information if desired.

A normal manner of providing an address or location on the Internet is in the form of a URL, whereby the providing means preferably should be able to provide such addresses or locations.

An interesting embodiment of the system is one wherein the transmitting means of the first computer is adapted to transmit additional information to the second computer, and wherein the second computer is adapted to transmit the computer code to the first computer if the second computer receives the additional information and to transmit the information requested if the second computer does not receive the additional information. In that manner, a first computer adapted to perform the method of the invention will identify itself vis-á-vis the server by the additional information where after the server will transmit the code. Other computers requesting the information will simply receive the information in the normal manner - each time the information is requested.

In the following, preferred embodiments of the invention are described for use in reducing the bandwidth required for transmitting Internet pages, home pages from a server to a client.

Brief description of the drawings

The objects and advantages of the invention will be apparent upon consideration of the following detailed description, taken in conjunction with the accompanying drawings, in which the reference characters refer to like parts throughout and in which:

FIG. 1 shows a computer network containing a client and a server system;

FIG. 2 shows a data process system suitable for practising methods and systems consistent with the present invention;

FIG. 3 depicts the data flow between the client, the server proxy 256 and the server; and

FIG. 4 shows an exemplary embodiment of script execution.

Detailed description of the preferred embodiments

Reference will now be made in detail to an implementation embodiments of the present invention, as illustrated in the accompanying drawings. Wherever possible, the same reference numbers will be used throughout the drawings and the following description to refer to the same or like parts.

Overview

Systems and methods consistent with the present invention use a server proxy 256 on a client system for intercepting or post-processing requests from a client process to the server process. Based on a set of rules, the server proxy 256 handles these requests locally by loading a resource template containing data and blocks of instructions which are executed as part of the resource retrieval. The server proxy 256 interfaces with the client and server processes and monitors requests that are transmitted from the client process to the server process. The system and method described herein allow users to create resource templates, which are registered with the server proxy 256, thereby

allowing the user to substitute the steps of contacting the server process and downloading the resource with the steps of loading the resource template present on the client and execute scripts inside the template.

The resource template contains static data and scripts (blocks of executable instructions) recognized by the server proxy 256 and executed while loading the resource template. The execution of the scripts result in dynamic data being inserted into specific, appropriate places in the resource template. Once all of the scripts have been executed, the resource is completely constructed and delivered to the client process by the server proxy 256.

When resource templates are registered with the server proxy 256, they are associated with a resource identity in a way consistent with the method of identifying resources employed by the client-server framework.

When monitoring requests made by the client process, the server proxy 256 examines the identity of the resource being requested. If the identity of the resource matches a registered resource template, the request is handled by the server proxy 256, which takes complete responsibility for delivering the resource to the client process. If the identity of the requested resource does not match a registered resource template, the request is transmitted to the server process and the response from the server is delivered to the client process.

Details of how resource templates are associated with resource identities may differ in systems consistent with the present invention and include (without limitation) associating a specific identity with a specific resource template and associate a partial identifier with a group of resource templates.

For example, a specific URL can be associated with a specific resource template as well as a partial URL (or base URL) be associated with a group of resource templates

in a namespace.

Detailed description

FIG. 1 shows, in more detail, an example of a client-server system interconnected through a network 100. In the example, a remote server system 122 is interconnected through network 100 to client system 120. Client system 120 includes conventional components such as a processor 124, memory 125 (e.g., RAM), a bus 126 which couples processor 124 and memory 125, a mass storage device 127 (e.g., a magnetic hard disk or an optical storage disk) coupled to processor 124 and memory 125 through an 1/0 controller 128 (connected to bus 126) and a network interface 129, such as a conventional modem or network adapter (connected to bus 126).

Server system 122 also includes conventional components such as a processor 134, memory 135 (e.g., RAM), a bus 136 which couples processor 134 and memory 135, a mass storage device 137 (e.g., a magnetic or optical disc) coupled to processor 134 and memory 135 through an I/O controller 138 (connected to bus 136) and a network interface 139 such as a conventional modem or network adapter (connected to bus 136). It will be appreciated from the description below that the present invention may be implemented in software, which is stored, for example, as executable instructions on a computer readable medium on the client, such as mass storage devices such as 127, or in memory 125.

It will be recognized by one skilled in the art that the network 100 shown in FIG. 1 may be any topology and operate under any protocol, including the Internet, private networks, wireless networks and internal computer structures, that allows either various computer systems or modules within a single computer to exchange information.

FIG. 2 shows a data processing system suitable for practising the present invention. The data system comprises a client system 250 and a server system 200. The client and

server systems 250, 200 have a number of components, including processors 251 and 201, respectively, memory 253 and 203, respectively, storage devices 242 and 202, respectively and network adapters 254 and 204, respectively. The client and the server are connected through a network (not shown in FIG. 2), which may be one of the types of networks mentioned above.

The client memory 253 contains the client program 255, the server proxy 256 and an operating system 257. When the client program 255 makes a request to the server system 200, the request is made to the operating system 257. The server proxy 256 contains an association 258 between a specific resource identifier and a resource template 259 stored in the storage device 252 on the client. The resource template 259 contains the static parts of the resource 206 and a script to download the dynamic part of the resource from the server process 205. The server memory 203 contains the server process 205 that the client communicates with. The mass storage device 202 on the server system 200 contains the resource 206 with the identity being associated with the resource template 258 in the server proxy 256. In addition, the server mass storage 202 contains the dynamic part of the resource 206 in a separate resource 207. Under normal operation, that is, without the server proxy 256 present on the client system 250, the client program 255 would request the data (resource) from the server process 205 by invoking the operating system 257 which would transmit the request over the network through the network adapters 254 and 204 and finally to the server process 205. In turn, the server process 205 would load the resource 206 from the mass storage 202 of the server system 200 and transmit the requested resource 206 to the client system 250 over the network through the network adapters 204 and 254. The operating system 257 on the client would then deliver the requested resource 206 to the client program 255. Using the server proxy 256, the resource request from the client program 255 is intercepted by the server proxy 256 before being transmitted through the network adapter 254 on the client system 250. The server proxy 256 loads the resource template 259 from storage 252 and executes the script contains therein (if one is present). This script may include instructions for downloading the resource 207 from the

server (which is downloaded as described above). Once downloaded, the script (or the server proxy 256) inserts the resource 207 into the resource template 259, completing the assembly of the resource which is then delivered to the operating system 257 by the server proxy 256. The operating system 257 completes the transaction by returning the locally generated resource to the requesting client program 255.

If large parts of the resource 206 change rarely, but a small part 207 of the resource changes frequently and the resource must be delivered in one piece to the client process, the introduction of the server proxy 256 and the resource template result is reduction of bandwidth usage when requesting the resource, since only the dynamic part of the resource 207 is actually transmitted over the network.

One skilled in the art will recognize that the resources 206 and 207 may be documents, media or even programs being executed on the server system 200. Resources downloaded by scripts in resource templates may be cached using a traditional cache scheme known to those skilled in the art.

Furthermore, it will be appreciated that the script may perform any computational task, including but not limited to dynamically constructing data such as images as well as downloading one or more resources from one or more servers and querying resources from local databases.

It will further be recognized by one skilled in the art that the resource template might as well be stored in memory, on a removable media such as a CD-ROM, in a database or on a local intranet server depending on the specific realization of the present invention.

FIG. 3 shows the flow of data for a request that matches a registered resource template and a request that does not.

If a client process 300 makes a request that does not match a registered resource template, the request is intercepted by the server proxy 301 which examines the request 303 and determines that no matching resource template is available and forwards the request to the server system 305 and consequently the server process 304. The server process 304 returns the requested resource, which is delivered to the client process 300 without interference of the server proxy 256.

If a request that is made by the client process 300 matches a registered resource template, the request is intercepted by the server proxy 301, the proxy server 301 process the request by loading the associated resource template 308 and processing the scripts inside the resource template 309. Finally the server proxy 303 delivers the requested resource to the client process 300.

In some embodiments of the present invention, the process of matching resource identities with resource templates may be implemented by associating a base URL (for example "http://www.netbloom.com/local") with a local directory (for example "C:\netbloom") on the hard disc of the computer and the using these base paths for mapping a subset of the URL's namespace to local files on the hard disc. With this approach, the server proxy 256 maps a request with the URL of "http://www.netbloom.com/local/dir/index.html" to a local file with the path "C:\netbloom\dir\index.html", loads the file and executes any scripts in the resource template before returning the resource to the client process. Likewise the server proxy 256 may map a single URL to a specific file in the file system.

The server proxy 256 may choose to forward the request to the server in case the URL was inside the namespace but no file is present in the local file system or it may return an error to the client process.

Another aspect consistent with the methods and systems consistent with the present invention is that the namespace handlers can be added to the server proxy 256 as

plug-ins. A namespace handler is responsible for mapping resource identities to local resource templates using a scheme similar to the exemplary namespace handler described above.

Plug-in namespace handlers compliant with a specific API can thus be easily added to the server proxy 256, allowing for mapping resource identities with resource templates that are located and maintained by separate systems, for example databases or proprietary applications.

The process of associating a resource with a local resource template may then be associating a resource or group of resources with a namespace handler and a set of parameters for this handler.

The example above would then comprise associating a base URL with a filesystem namespace handler specifying the root directory of the mapping. Likewise, a base URL could be associated with a CD-ROM namespace handler mapping the base URL to the root of the CD-ROM drive on the client and requiring the CD present in the drive to have a specific label.

In one embodiment of the present invention, scripts are placed inside the resource templates and marked by special tags. The processing of the scripts in a resource template, once it is loaded, comprises identifying the script blocks intersected in the resource template and concatenating the static parts of the resource template with the output generated from running the scripts.

FIG. 4 shows an exemplary embodiment of this aspect, where the resource template 400, an HTML document, comprises of two static parts 401 and 403 and a script 402. When processing the script 402 the server proxy 256 concatenates the static part 401 with generated output "Hello World" from executing the script 402 and the second static part of the resource template, yielding the resource 404 which is delivered to the

client process.

Besides control structures such as conditionals and loops, the server proxy 256 may provide functionality for downloading data from servers using standard communication protocols such as HTTP, FTP and secure communication protocols such as HTTPS, functionality for adding resources to existing namespaces and database functionality.

In addition, the scripting language might provide interfaces to standard object technologies such as COM, JavaBeans or CORBA making it possible to dynamically extend the functionality available to the scripts being executed as part of the resource template by installing COM objects, JavaBean components or CORBA services on the local system.

It will be recognized by one skilled in the art that programming language used in the script blocks of the resource templates may be any existing programming language or a language specific to the embodiment of the invention. In addition, the language may be interpreted or compiled.

Furthermore, it will be appreciated that the script and the static parts of the resource template may be separated into separate files and that a resource template containing only script or only static data are valid resource templates. Finally, the resource templates may be cached in memory in a compiled form in order to speed up subsequent requests for the resource handled by the specific resource template.

Methods and systems consistent with the present invention may group resource templates or namespaces in applications, maintaining private application and session state for each application and active user session. For example, if the client process is a web browser and a server process is a web server, an application may be defined containing a group of namespaces. Alternatively, applications may be grouped by domains.

When the first URL inside one of the applications namespaces or a specific domain is requested, the application is initialized and a session is created. The application may load a configuration, which is exposed to the scripts being executed in the resource templates belonging to the namespaces of the application.

Likewise, a session state may be exposed to the scripts allowing the scripts in the resource templates belonging to the application to accumulate and share information during the session and use this information when processing requests.

The session is terminated when the client process dies or alternatively when no request has been made for the particular application for a given period of time. The server proxy 256 may be running several applications simultaneously and the data for each application may be isolated for security reasons.

Another aspect of systems and methods consistent with the present invention is that resource templates can be installed permanently on the local system from different sources, such as CID-ROM's, downloads and the like.

If the server proxy 256 is enabled on the client, the process of installing resource templates may comprise copying the resource templates to the hard disc of the client and registering the templates with the server proxy 256.

Registering templates may be done by adding the definitions of the resource templates and their associated resource identifiers to a configuration file, the system registry, a database or some similar way known to one skilled in the art. In yet another aspect of systems and methods consistent with the present invention is that the resource templates can be downloaded dynamically and cached by the server proxy 256.

By adding information to the request being sent to the server, the server proxy 256 can make its presence on the client clear to the server.

If capable, the server may respond by sending a resource template to the browser rather than sending the actual resource. The server indicates the server proxy 256 that the data being transmitted is actually a resource template and not the actual resource by adding information to the response.

When the resource template is downloaded, it is executed as described above completing the construction of the resource possibly downloading additional data. The server proxy 256 caches the resource templates and subsequent requests for the resource will be handled by the cached resource template rather than the server.

The caching scheme could be any traditionally timestamp-based scheme known to one skilled in the art.

Furthermore, the functionality for cached resource templates may be restricted to resource templates being installed on the client for security reasons.

The foregoing description of an implementation of the invention has been presented for purposes of illustration and description. It is not exhaustive and does not limit the invention to the precise form disclosed. Modifications and variations are possible in light of the above teachings or may be acquired from the practising of the invention. For example, the described implementation includes software but systems and methods consistent with the present invention may be implemented as a combination of hardware and software or in hardware alone. The invention may be implemented with both object-oriented and non-object-oriented programming systems. Additionally, although aspects of the present invention are described for being stored in memory, one skilled in the art will appreciate that these aspects can also be stored on other types of computer-readable media, such as secondary storage devices, like hard disks, floppy disks, or CD-ROM; a carrier wave from the Internet or other propagation medium; or other forms of RAM or ROM, the scope of the invention is defined by the claims and their equivalents.

Thus are provided methods, systems and devices for providing information from a server to a client. One skilled in the art will appreciate that the present invention can be practised by other than the described embodiments, which are presented for purposes of illustration and not limitation, and the present invention is limited only by the claims that follow.

Claims

1. A method for providing information at a first computer, the method comprising:

at a first point in time:

providing, at the first computer, information identifying the information desired;

transferring the information from the first computer to a second computer;

transferring processor executable computer code from the second computer to the first computer,

at a point in time subsequent to the first point in time:

providing, at the first computer, the identifying information,

executing in or at the first computer the computer code.

2. A method according to claim 1, wherein, at the first point in time, information is transferred from the second computer to the first computer, part of the transferred information being stored locally at or in the first computer.

3. A method according to claim 2, wherein the execution of the computer code comprises retrieving the part of the information from the local store and requesting, from the address or location, one or more other parts of the information.

4. A method according to claim 3, wherein the execution of the computer code comprises determining whether information required is stored in the local store.

5. A method according to claim 2, wherein, at the first point in time, information identifying the stored information is stored at or in the first computer.

6. A method according to claim 1, wherein the execution of the computer code comprises:
providing, at the first computer, information relating to a second address or location, and
transferring information from the second address or location to the first computer.

7. A method according to claim 2, wherein, at the first point in time, part of the information is stored locally at or in the first computer and wherein, at the subsequent point in time, the part of the information is retrieved from the local store and one or more other parts of the information is transferred from the second address or location to the first computer.

8. A method according to claim 2, wherein the execution of the computer code comprises determining whether information stored in the local store is outdated and, in that situation, requesting updated information from the second computer.

9. A method according to claim 1, wherein the execution of the computer code comprises storing at least part of the transferred information locally at or in the first computer or retrieving information therefrom, the storing or retrieval being performed in a local database defined and/or maintained by execution of processor executable computer code transferred from the second computer.

10. A method according to claim 1, wherein the information relating to the address or location is a URL.

11. A method according to claim 1, wherein the information to be provided is a home page, a web site, or one or more parts thereof, the information comprising a number of independent resources, and wherein the address or location thereof is on or at a second computer holding the independent

resources.

12. A method according to claim 11, the method comprising identifying, at the first computer, parts of the information which are present at the first computer, requesting, at the second computer, parts of the information not present at the first computer,
retrieving the information required stored in or at the first computer,
combining the information required and providing it to a user of the first computer.

13. A method according to claim 12, wherein the process of transferring the information relating to the address or location to the second computer comprises transmitting additional information to the second computer, and wherein the process of transmitting the processor executable computer code to the first computer is conditioned on the second computer receiving the additional information, the second computer otherwise transmitting the information required to the first computer.

14. A method according to claim 1, the method comprising:
in each of a plurality of first computers:
at a respective first point in time:
providing, at the first computer, information relating to an address or location of the information desired,
transferring the information relating to the address or location from the first computer to a second computer,
transferring processor executable computer code from the second computer to the first computer,
at a second respective point in time subsequent to the respective first point in time:
providing, at the first computer, the information relating to the address or location,
executing in or at the first computer the computer code,

where the information relating to the addresses or locations provided relates to the same information and wherein the processor executable computer code transferred is adapted to provide the same effect in the first computers when executed.

15. A method according to claim 14, wherein, in each first computer at the respective first point in time, information is transferred from the second computer to the first computer, the same parts of the transferred information being stored locally at or in each first computer.

16. A method according to claim 15, wherein the execution of the computer code comprises, in each first computer, retrieving the parts of the information from the local store and requesting, from the address or location, the same one or more other parts of the information.

17. A system for providing information from a first computer to a second computer, the system comprising a first and a second computer where:
the first computer comprising:

a mechanism constructed and adapted to provide information relating to an address or location of the information desired,

a mechanism constructed and adapted to transfer the information from the address or location to the first computer to the second computer,

a mechanism constructed and adapted to receive, from the second computer, processor executable computer code, and

a mechanism constructed and adapted to execute the computer code,

a mechanism constructed and adapted to determine whether the address or location is provided by the providing a mechanism constructed and adapted lo the first or a subsequent time and for, if the address or location is provided a subsequent time, operating the executing means to execute the computer code,

the second computer comprising:

a mechanism constructed and adapted to receive the address or location,

a mechanism constructed and adapted to transmit the computer code to the first computer.

8. A system according to claim 17, wherein:

the first computer comprises storage means,

the receiving mechanism comprises a mechanism constructed and adapted to receive information and to store at least part of the received information in the storage means,

the executing mechanism being adapted to retrieve at least part of the stored information when executing the computer code.

19. A system according to claim 18, wherein the executing mechanism is adapted to, while executing the computer code, define and/or maintain a database held by the storage means, and wherein the storing means comprise a mechanism constructed and adapted to provide information relating the address or location with a location of the stored information.

20. A system according to claim 18, wherein the executing means are adapted to, while executing the computer code, determine whether all or part of the stored information to be retrieved is outdated and to, in that situation, request updated information from the second computer.

21. A system according to claim 17, wherein the providing mechanism is adapted to provide an address or location in the form of a URL.

22. A system according to claim 17, wherein the transmitting mechanism of the first computer is adapted to transmit additional information to the second computer, and wherein the second computer is adapted to transmit the computer code to the first computer if the second computer receives the additional information and to transmit the information requested if the second computer

does not receive the additional information.

23. A computer program comprising computer program code means adapted to perform the steps of claim 1 when said program is run on a computer.

24. A computer program as claimed in claim 23 embodied on a computer readable medium.

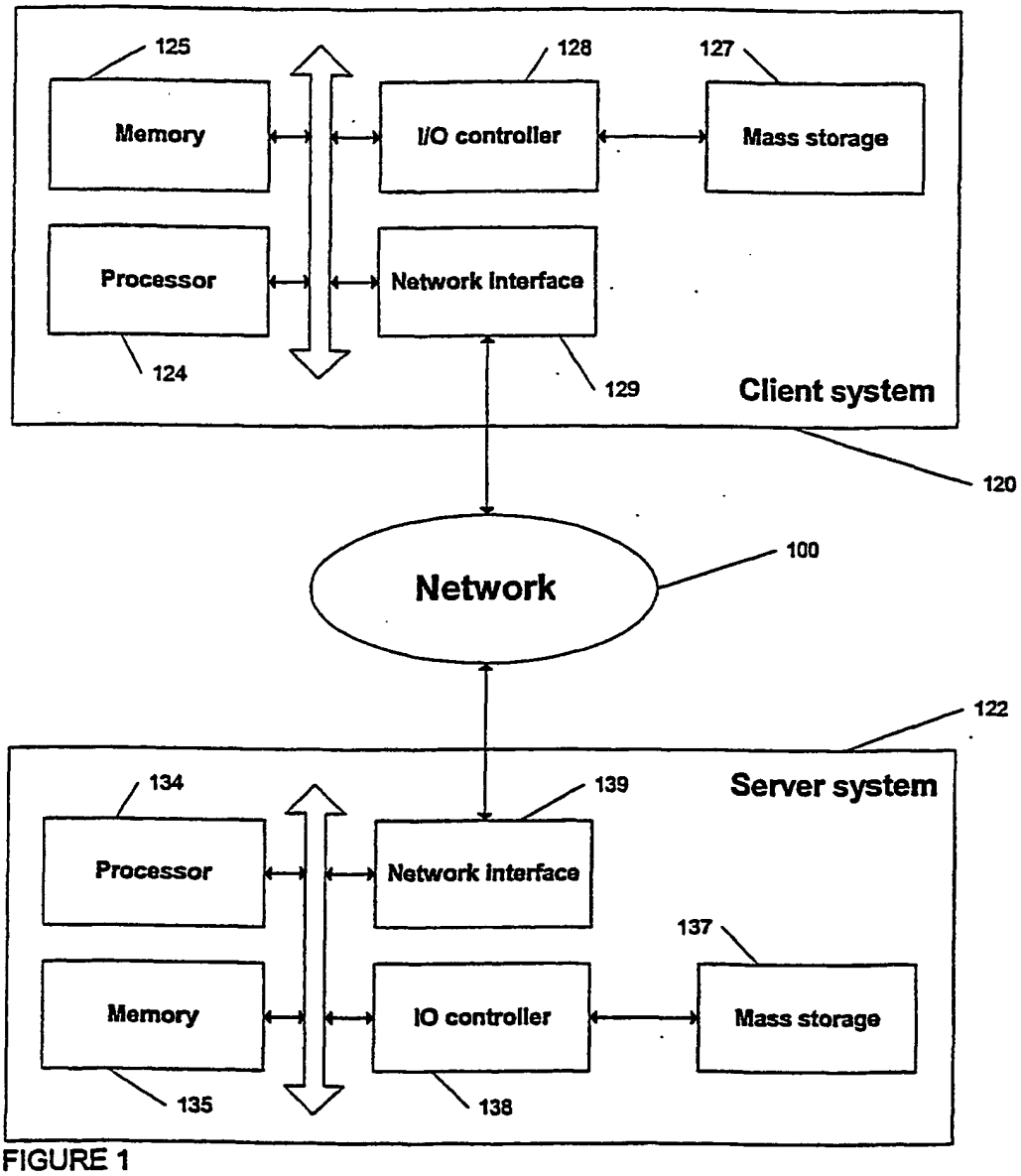

125
128
127
Memory
I/O controller
Mass storage
Processor
Network Interface
124
129
Client system
120
Network
100
122
Server system
134
139
Processor
Network Interface
137
Memory
IO controller
Mass storage
135
138

FIGURE 1

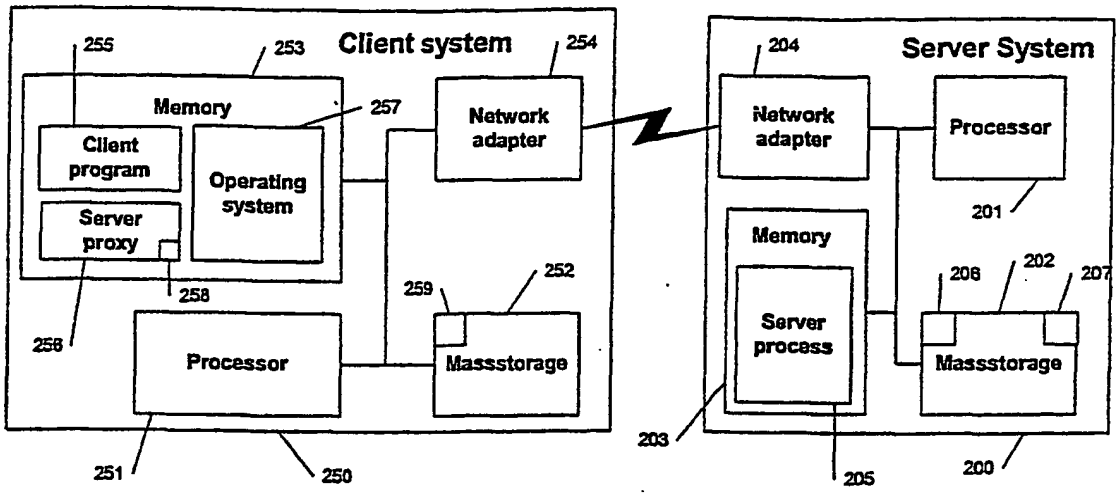

Client system
255
253
254
Memory
257
Client program
Operating system
Network adapter
Server proxy
258
258
259
252
Processor
Massstorage
251
250
Server System
204
Network adapter
Processor
201
Memory
Server process
208
202
207
Massstorage
203
205
200

FIGURE 2

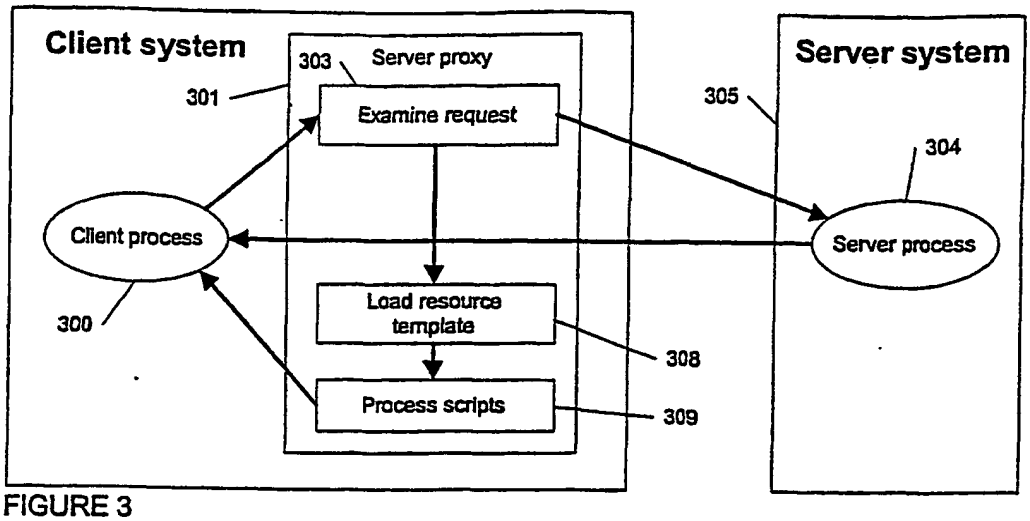

Client system
303
Server proxy
301
Examine request
305
Server system
304
Client process
Server process
300
Load resource template
308
Process scripts
309

FIGURE 3

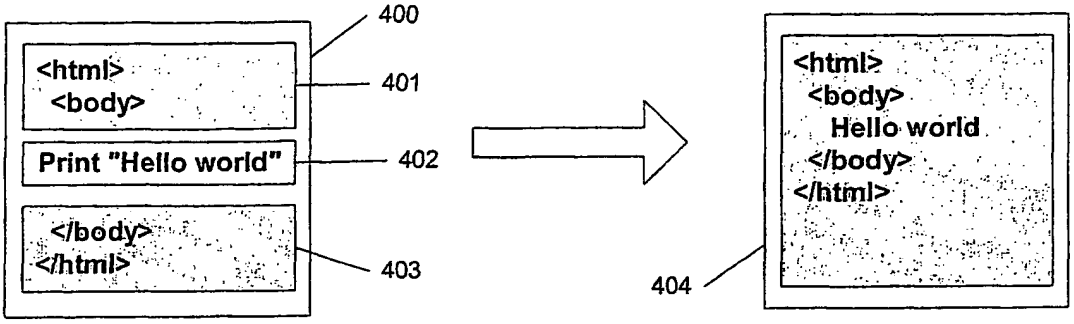


FIGURE 4